A Few Words About MathML
An introduction to MathML, and specifically an explanation as to its use in web education.

Math is a complicated language: There are hundreds of symbols and signs used to create mathematical expressions. In fact, very few computer systems come equipped to display all of them. Many web sites, including Connexions, are beginning to use a markup language called **MathML** to display math expressions and equations so that they may be viewed by any visitor.

Why MathML?

Before MathML was introduced, most web pages used images of math equations rather than markup. There are several problems with this simple approach: Images can't be read by screen readers for disabled users and images do not scale well for presentation purposes, just to name a few. And while some Windows machines come with a limited number of fonts to display mathematical symbols, these may not correspond to the same symbols used on Unix and Mac OS systems. Finally, typesetting math can be downright impossible over the Internet without the use of tables and style sheets to individually position each part of the expression.

MathML solves all of these problems by giving the user complete control over an expression's layout and content. Each element in an expression is tagged as a number, letter, operator, or symbol, then can be placed in tags that denote the sequence of operations applied. Alternately, the expression can be built piece-by-piece, symbol-by-symbol, using tags to explicitly control the way the math will be presented. When users view a web site with embedded MathML content, their web browser reads the tags and displays the content appropriately.

Presentation MathML Versus Content MathML

A comparison of the two "syntaxes" of MathML: Presentation and Content. This module weighs the pros and cons of each syntax in a variety of teaching situations.

There are two "flavors," or synataxes, of MathML: **Presentation MathML** and **Content MathML** (or P-MathML and C-MathML, respectively). Both are valid markup and follow the same basic set of rules. P-MathML allows authors to have complete control over how an expression looks: Its size, color, which symbols are used, and the exact position of each object and element. Expressions written in C-MathML, on the other hand, maintain their sense of meaning by using tags that denote operations and how they should be applied. C-MathML preserves the **semantics** of an expression.

If this seems confusing, just remember that P-MML describes how an expression **looks**, while C-MML describes what an expression **does**.

There are drawbacks to using either P-MML or C-MML exclusively. Consider the following example which demonstrates how an expression might be read back by a screen reader (a tool used by the visually impaired to read on-screen text), depending on whether it was written using P-MathML or C-MathML.

Equation	**Content MathML**	**Presentation MathML**

Equation	**Content MathML**	**Presentation MathML**
—	The integral of A with respect to X is equal to the second derivative of the quantity three times B to the fourth power with respect to X.	Integral sign, A, d, X, equals, d, squared, quantity, three, B, fourth power, end quantity, divided by, d, X, squared.

The semantics of Content versus Presentation MathML

Obviously, C-MML would be much more useful in this case; P-MML cannot convey any useful information about the equation except for how to write it down.

There are also cases in which P-MML is the more appropriate syntax of MathML to use. Consider a class of first-year calculus students learning about the derivative. Their instructor needs to write a math worksheet for the class in MathML. There are many ways to write the derivative of ...

-
- F/dx
- F-dot

... but this class only knows prime notation at the moment. If C-MML were used to build the worksheet, there is a chance that some students' browsers may display the derivative in the wrong form. Using P-MML, the instructor can specify which derivative notation to use.

It is important to consider the audience when authoring MathML. If accessibility and content are not an issue, then P-MML offers a very easy-to-understand syntax for sharing mathematical equations. As a rule of thumb, Content-MathML should be used whenever possible, since it allows expressions to be viewed on a wide range of browsers, platforms, languages, and regions.

Introduction
A brief summary of Content MathML, including an example and its use.

Content MathML is a markup language used to describe mathematical expressions by correctly preseving their **semantics**. Rather than focusing on the way an equation or expression looks (like Presentation MathML), Content MathML (or C-MML) aims to preserve the **meaning** of the expression so that it can be interpreted by anyone. It does this by marking up each object with special tags that denote which operations to apply and in which order to apply them. The mathematical expression can then be view by anyone with a MathML-enabled browser with the correct math fonts installed.

If you're familiar with XML or HTML, then you've already seen markup languages in action. Markup languages are used for presenting documents such as web sites according to certain structural rules. In the case of C-MML, the tags contain all the structural information about the expression.

Below is the classic equation $2 + 2 = 4$ represented in MathML.

```
<m:math>
  <m:apply>
    <m:eq/>
    <m:apply>
      <m:plus/>
      <m:cn>2</m:cn>
      <m:cn>2</m:cn>
    </m:apply>
    <m:cn>4</m:cn>
  </m:apply>
</m:math>
```
$2 + 2 = 4$ expressed in Content MathML

Take a look at the above markup code: It may seem like a lot of information and tagging to represent something as easy-to-write as $2 + 2 = 4$. Remember, though, that C-MathML is interested in conveying the semantics of the expression, not just displaying symbols on the screen. Math is also a very explicit language, so MathML expressions must be somewhat long in order to convey meaning. Bigger and more complex equations benefit from the organization and explicit tagging of Content MathML, but for the sake of providing a simple example we will stick with $2 + 2 = 4$.

MathML was designed to be built by special programs called **equation editors**, and not written by hand. While equation editors are great tools to use while learning the language, they are not necessary to build an effective MathML expression. In fact, many equation editors produce semantically **incorrect** MathML. It's important to always double-check any MathML markup created with an equation editor, especially to look for errors in the order of operations.

Content MathML Tags

A comprehensive explanation of the most commonly-used Content MathML tags. Provides an basic, step-by-step explanation as to how Content MathML expressions are built using these tags.

Let's take a close look at $2+2=4$ expressed in Content MathML. From it, we can identify the most common tags used in C-MML expressions.

Example:
Writing $2+2=4$ in Content MathML

```
<m:math>
  <m:apply>
  <m:eq/>
    <m:apply>
    <m:plus/>
      <m:cn>2</m:cn>
      <m:cn>2</m:cn>
    </m:apply>
  <m:cn>4</m:cn>
  </m:apply>
</m:math>
```

There are a few things you should note immediately about the code above. Every tag (or **element**, as they are formally known) is contained in brackets and begins with a "m:" to signify that it is a MathML tag. We say that this "m:" defines the **namespace** of the element. Namespaces are used in web markup languages to keep track of which language (such as XHTML or QML) is being "spoken." Also notice that the equation begins with the `math` element, which denotes the beginning of every MathML expression.

Like other web-based markup languages, MathML tags have corresponding **end tags** which mark the end of a section-- in this case, the end of an operation or modifier. Every end tag is essentially the same as it's start tag, but with a forward slash ("/") character in front of the name.

There are over a hundred different Content MathML tags, but only a few that are used in the majority of expressions. Remember, you can always refer to the W3C's Content MathML page to look up a tag if you can't remember it.

Tokens: cn, ci, and csymbol

Each number, variable, and symbol has to be explicitly marked up in a given MathML expression. There are three different elements used to mark up these operands.

- `cn`
- `ci`
- `csymbol`

cn

The **cn** element is used to denote an explicit number, as opposed to a variable or symbolic representation of a number. Values such as 1, -5000, and 3.14159 are all marked up with `cn` tags.

In our simple example, $2 + 2 = 4$, there are three numbers marked up with `cn` tags.

```
<m:math>
  <m:apply>
  <m:eq/>
    <m:apply>
    <m:plus/>
      <m:cn>2</m:cn>
      <m:cn>2</m:cn>
    </m:apply>
  <m:cn>4</m:cn>
  </m:apply>
</m:math>
```

`ci`

The **ci** element is used to markup variables. Any non-explicit representation of a number, such as X, F, or P_2, is contained in a `ci` element.

If we wished to change our example expression into an algebraic relation, say $x + 2 = 4$, then we would only need to make one small change to the Content MathML markup.

```
<m:math>
  <m:apply>
  <m:eq/>
    <m:apply>
    <m:plus/>
      <m:ci>x</m:ci>
      <m:cn>2</m:cn>
    </m:apply>
  <m:cn>4</m:cn>
  </m:apply>
</m:math>
```

While the **csymbol** is a valid Content MathML construct, its use is fairly uncommon and therefor it will not be discussed in this module.

Apply

Now that we understand how to markup operands, we need to know how to relate and operate upon them to form an expression. The most fundamental tag for writing Content MathML is the **apply** tag, which governs the way operations and modifiers are placed in the expression. Take another look at the example above, and you might notice that there are two `apply` tags. The `apply` tag is MathML's way of indicating that an operation or relation should be introduced.

The first child of an `apply` tag is the operation or relation we wish to implement. Colloquially, a **child** is any element directly within (or "under") an element.

```
<m:math>
  <m:apply>
  <m:eq/>
    <m:apply>
    <m:plus/>
      <m:cn>2</m:cn>
      <m:cn>2</m:cn>
    </m:apply>
  <m:cn>4</m:cn>
  </m:apply>
</m:math>
```

In the code block above, the `apply` tag is highlighted in bold and the `apply` tag's first child element is italicized. The `apply` tag will always understand its first child to be the operation or relation. The operation in this case is `plus` (addition).

The second child of the `apply` tag is always the first element being acted upon. There may be even more children depending on what kind of operation is being applied. It wouldn't make any sense to apply sine to two numbers but it **does** make sense of apply addition to two numbers. For our example, there are two child elements after the `plus` tag, both of which are `cn` elements containing "2". So the instruction reads "apply addition to two and two," or $2 + 2$

It can be difficult to understand the way that `apply` tags work since MathML doesn't "read" like a typical mathematical expression. Again, remember that Content MathML is focused on applying operations in order to convey semantics. If we were to simply transcribe our Content MathML example onto paper as operations and numbers, it would look something like this:

$= (+22)(4)$

Equations written in this way are said to be in **prefix** notation (or **Polish notation**). It's may seem very counter-intuitive, but computers handle this notation a lot faster than "normal" (or **infix**) notation.

To expand our understanding of the `apply` tag in Content MathML, let's look at the topmost `apply` tag and its children.

```
<m:math>
  <m:apply>
  <m:eq/>
  <m:apply>
    <m:plus/>
      <m:cn>2</m:cn>
      <m:cn>2</m:cn>
    </m:apply>
  <m:cn>4</m:cn>
  </m:apply>
</m:math>
```

The first child of `apply` is the `eq` element, which corresponds to equality ("="). The second child, however, is another `apply` tag. This means that **everything inside this second `apply` tag is the first argument**. The third direct child of the first apply tag is a `cn` containing the number 4. This means that "whatever is in the second `apply` tag is equal to four."

Relations and Operators

Mathematical expressions usually involve at least one **relation**, such as equality ("=") or one number being greater than the other ("<" or ">"). Content MathML offers an extensive library of relational elements. The `<eq/>` relation is highlighted in our example expression below.

```
<m:math>
  <m:apply>
  <m:eq/>
    <m:apply>
    <m:plus/>
```

```
      <m:cn>2</m:cn>
      <m:cn>2</m:cn>
    </m:apply>
  <m:cn>4</m:cn>
  </m:apply>
</m:math>
```

Notice that the `<eq/>` element does not have an end tag. This is because relations and operations are generally declared using an **empty tag**. It doesn't "contain" anything, so it simply terminates itself with a forward slash at the end of the element.

There are several commonly-used Content MathML relation elements. These include:

- `eq` ($\mathrm{a} = \mathrm{b}$)
- `neq` ($\mathrm{a} \neq \mathrm{b}$)
- `gt` ($\mathrm{a} > \mathrm{b}$)
- `lt` ($\mathrm{a} < \mathrm{b}$)
- `geq` ($\mathrm{a} \geq \mathrm{b}$)
- `leq` ($\mathrm{a} \leq \mathrm{b}$)

Similarly, we usually have at least one **operator** in a given mathematical expression. Some of the most common operators used in Content MathML include:

- `plus` ($a + b = \mathrm{c}$)
- `minus` ($a - b = \mathrm{c}$)
- `divide` ($\frac{a}{b} = \mathrm{c}$)
- `times` ($ab = \mathrm{c}$)
- `power` ($a^b = \mathrm{c}$)
- `root` ($\sqrt{ab} = \mathrm{c}$)

Content MathML Cookbook
A series of code examples explaining how to build basic algebra, calculus, trigonometry, and differential expressions using Content MathML

Content MathML, like any language, is difficult to "learn" completely. Becoming comfortable with the semantics and structure of Content MathML is a lot easier than memorizing every possible element

This MathML "cookbook" provides you with the "ingredients" (or helpful examples) to begin writing your own MathML expressions. Namespaces are not included, so please be sure to include an appropriate namespace if combining these snippets with other XML languages.

Algebra

$$2 + 2 = 4$$

```
<math>
<apply>
  <eq/>
  <apply>
    <plus/>
    <cn>2</cn>
    <cn>2</cn>
  </apply>
  <cn>4</cn>
</apply>
</math>
```

$$x^2 + 2x + 3 = (x - 1)(x + 3)$$

```
<math>
<apply>
  <eq/>
  <apply>
    <plus/>
    <apply>
      <power/>
      <ci>x</ci>
      <cn>2</cn>
    </apply>
    <apply>
      <times/>
      <cn>2</cn>
      <ci>x</ci>
    </apply>
    <cn>3</cn>
  </apply>
  <apply>
    <times/>
    <apply>
      <minus/>
      <ci>x</ci>
      <cn>1</cn>
    </apply>
    <apply>
      <plus/>
      <ci>x</ci>
      <cn>3</cn>
    </apply>
  </apply>
</apply>
</math>
```

$$x = \frac{c-b}{a}$$

```
<math>
<apply>
  <eq/>
  <ci>x</ci>
  <apply>
    <divide/>
    <apply>
      <minus/>
      <ci>c</ci>
      <ci>b</ci>
    </apply>
    <ci>a</ci>
  </apply>
</apply>
</math>
```

$$\sqrt[3]{b} = \log_2 a$$

```
<math>
<apply>
  <eq/>
  <apply>
    <root/>
    <degree>
      <cn>3</cn>
    </degree>
    <ci>b</ci>
  </apply>
  <apply>
    <log/>
    <logbase>
      <cn>2</cn>
    </logbase>
    <ci>a</ci>
  </apply>
</apply>
</math>
```

Trigonometry

$$\sin^2(\theta) + \cos^2(\theta) = 1$$

```
<math>
  <apply>
    <eq/>
    <apply>
      <plus/>
      <apply>
        <power/>
        <apply>
          <sin/>
          <ci>?</ci>
        </apply>
        <cn>2</cn>
      </apply>
      <apply>
        <power/>
        <apply>
          <cos/>
          <ci>?</ci>
        </apply>
        <cn>2</cn>
      </apply>
    </apply>
    <cn>1</cn>
  </apply>
</math>
```

Calculus

$$\operatorname*{limit}_{h\to 0} \frac{f(x+h)-f(x)}{h}$$

```
<math>
  <apply>
    <limit/>
    <bvar>
      <ci>h</ci>
    </bvar>
    <lowlimit>
      <cn>0</cn>
    </lowlimit>
    <apply>
      <divide/>
      <apply>
        <minus/>
        <apply>
          <ci>f</ci>
          <apply>
            <plus/>
            <ci>x</ci>
            <ci>h</ci>
          </apply>
        </apply>
        <apply>
          <ci>f</ci>
          <ci>x</ci>
        </apply>
      </apply>
      <ci>h</ci>
    </apply>
  </apply>
</math>
```

$$F(x) = \int_a^x f(t)\,\mathrm{d}\,t$$

```
<math>
  <apply>
    <eq/>
    <apply>
      <ci>F</ci>
      <ci>x</ci>
    </apply>
    <apply>
```

```
      <int/>
      <bvar>
        <ci>t</ci>
      </bvar>
      <lowlimit>
        <ci>a</ci>
      </lowlimit>
      <uplimit>
        <ci>x</ci>
      </uplimit>
      <apply>
        <ci>f</ci>
        <ci>t</ci>
      </apply>
    </apply>
  </apply>
</math>
```

Differential Equations

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(f(y)\right)$$

```
<math>
<apply>
  <diff/>
  <bvar><ci> x </ci></bvar>
  <apply><ci> f </ci>
    <ci> y </ci>
  </apply>
</apply>
</math>
```

$$\frac{\partial^2 f(x,y)}{\partial x\,\partial y}$$

```
<math>
  <apply>
    <partialdiff/>
    <bvar>
      <ci>x</ci>
    </bvar>
    <bvar>
      <ci>y</ci>
    </bvar>
    <apply>
      <ci type="function">f</ci>
      <ci>x</ci>
      <ci>y</ci>
    </apply>
  </apply>
</math>
```

Introduction
Reference Manual for the MathML Editor

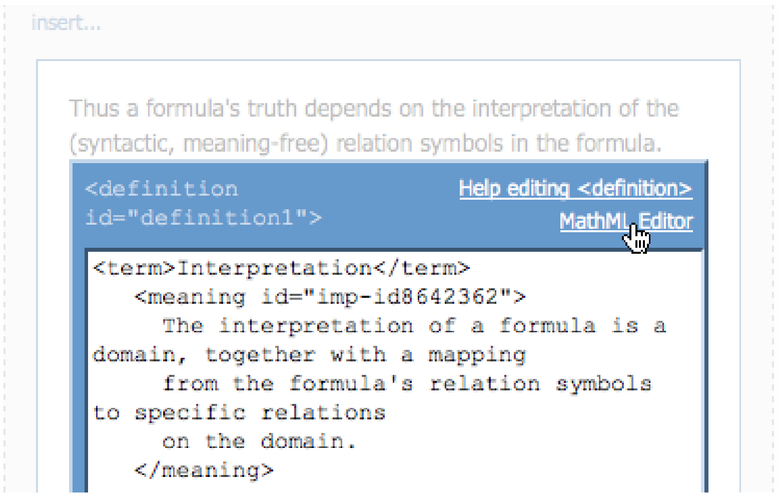
Math Editor Features

This module explains how to open the Math Editor, create math, edit existing math, and keyboard shortcuts. There is also a separate tutorial page with examples showcasing the features.

At the end of this module are nuances and limitations of the editor. Please, let us know which ones you'd really like to see incorporated!

Opening the Editor

When editing a Module using using Mozilla's Firefox browser, click on a part of the module to open a blue editing box. On the top-right hand side of the box is a "MathML Editor" link which will open up the editor.

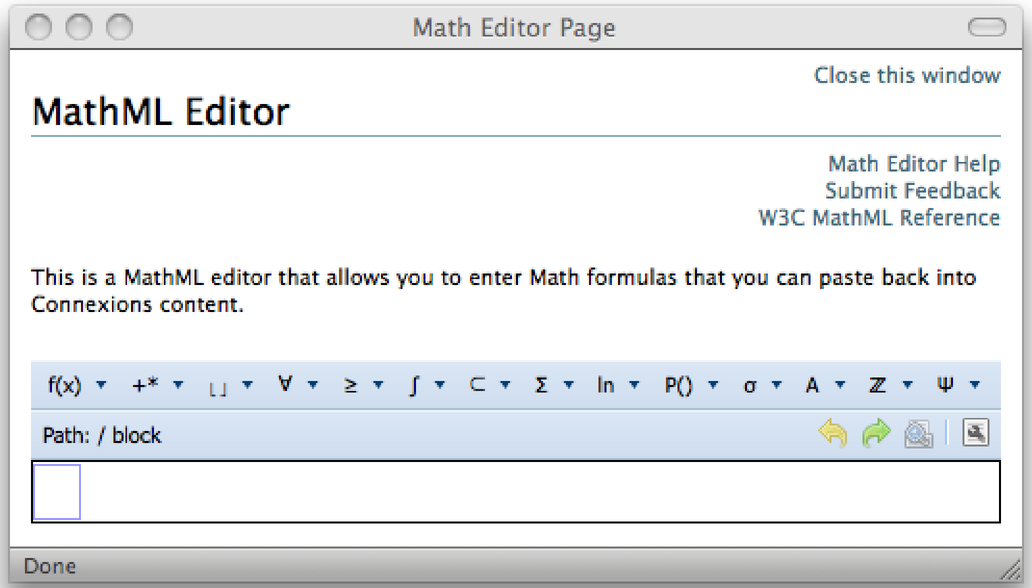


Thus a formula's truth depends on the interpretation of the (syntactic, meaning-free) relation symbols in the formula.

```
<definition
id="definition1">
```

```
<term>Interpretation</term>
   <meaning id="imp-id8642362">
     The interpretation of a formula is a
domain, together with a mapping
     from the formula's relation symbols
to specific relations
     on the domain.
   </meaning>
```

Begin editing and in the top-right corner is a MathML Editor link



This is a MathML editor that allows you to enter Math formulas that you can paste back into Connexions content.

Once clicked, a popup window will appear containing the Math Editor

Using the Popup Window

The popup window can remain open while editing a module and can even stay open while editing several modules. At any point one can close the window, but the contents of the editor will be lost.

User Interface

The Editor has 4 main sections, detailed below. The toolbar provides a way to insert new operations, a navigation tree to show where the cursor is located, and standard buttons for undo, preview, and source editing.

The main editing area is located below the toolbar and contains the math that is being edited.

Toolbar

The toolbar contains a row of buttons representing categories of different mathematical operations. These are enabled when something is selected in the editing area. **Note:** The editor does not infer multiplication and addition. See Nuances for how to insert next to existing math by wrapping existing math. Explain the different sections, when it's enabled, how things get inserted, and Keyboard Entry for things.

Menu
Clicking a category in the top row of the toolbar will open a menu of mathematical operations to insert. To the right of each operation is a name that can be entered from the keyboard at the cursor position in the main editing area. For example, instead of using the toolbar to enter the symbol for pi, the user could enter "`pi`" or "`<pi/>`" (the MathML version of pi) in the cursor and press the Enter key.

Path
Shows the path to the cursor location. Math is organized in a tree-like hierarchy (see Navigating Math) and the path represents where in the tree the cursor currently is. The path (and context) are important because they define what can be inserted and where it will go.

Undo/Redo
These buttons allow the user to undo an operation such as deletion or insertion. See Keyboard Shortcuts for details on using these features from just the keyboard.

Preview
Shows what math will look like when module is published. To resume editing, one must click the Preview button a second time.

View Source
Math in Connexions is represented in an XML format known as MathML. Clicking the View Source button will allow editing of the raw MathML.

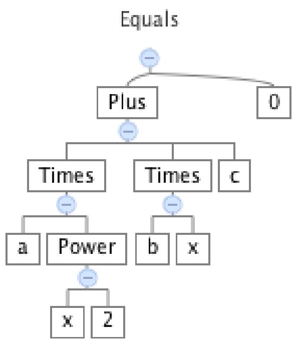
Editing Area

This is the main area for creating math. It begins empty, but math can be pasted directly in here from Connextions. The tutorials contain instructions on moving math from Connexions to the math editor and back. The editing area is the most important part of the editor and as several subsections, outlined below:

- Math is structured like a tree.
- Colors are used in this area to denote required information and contextual clues.
- Content vs Presentation Math discusses the two different types of math the editor supports.
- The cursor is discussed in detail below, including navigation and different editing modes.
- Since the exact location of the cursor may at times be ambiguous, the context provides visual cues.
- Keyboard strokes are discussed in detail.
- Finally, empty blocks are discussed below.

Math Tree

Math in the editor is structured like a tree. It can be thought of as removing the precedence rules and just having parentheses. For example, the formula "`a*x^2+b*x+c=0`" which is displayed (using the editor) as $ax^2 + bx + c = 0$ and as a tree would look like [link]. The equal sign has the least precedence and so is on the top. Similarly, x binds tighter to 2 through the `power` operation than to a through the `times` operation.



$ax^2 + bx + c = 0$
as a tree

Colors

Color notation (legend)

$$\boxed{\text{f}} = \begin{array}{lll} x_1 & \text{if} & x < 0 \\ x_2 & \text{if} & \square > 0 \\ \square & & \text{otherwise} \end{array}$$

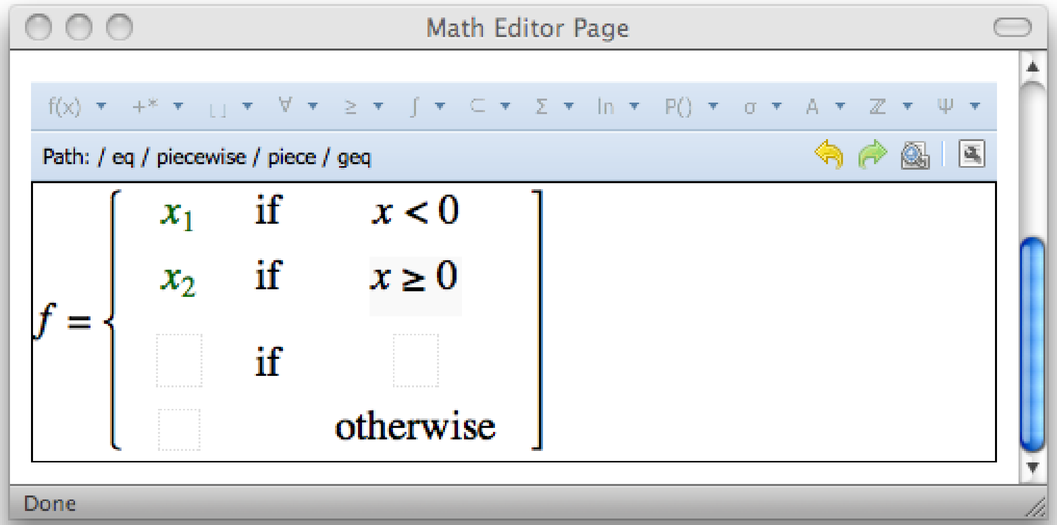
- $\boxed{\text{f}}$: The location where text is currently being entered is represented as a box with a blue border (see Text Input for more information on how to enter math).
- x and x_1 : Content MathML is represented in black while Presentation MathML is in a dark green (See Content vs. Presentation for editing Presentation MathML.
- $x < 0$: The cursor context (when the cursor is next to a complex expression) is represented by having a gray background. See Context for details.
- $\square$ and $\square$: Empty blocks that need to be filled are denoted with a yellow background and optional blocks that can be filled but do not need to be filled are transparent with a dotted border. See Blocks for details.
- x_2 : The current selection is denoted by a light blue background. See Copy and Paste back to Modules for details.

Content vs. Presentation

There are two subsets of the MathML language; Content MathML and Presentation MathML. Content, as the name implies, focuses on expressing operations like addition, integration, matrices, etc. Presentation focuses on how precisely math is displayed and contains elements like tables and subscripts.
Comparing Content and Presentation MathML

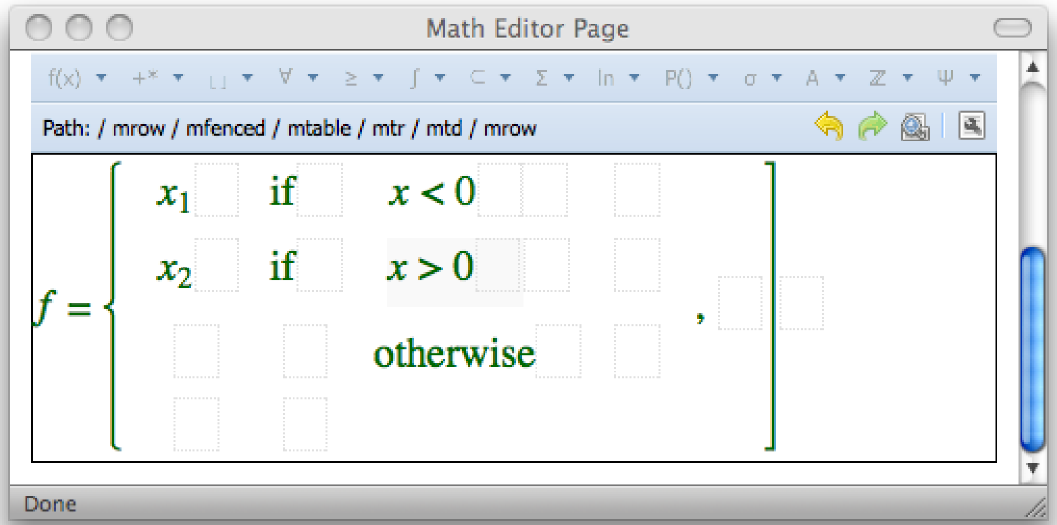
Content MathML typically has fewer places to enter information and navigation is simpler, and Presentation MathML allows the user to tweak the way formulas are presented and is used by OCR and import software.

Content MathML


Math Editor Page
f(x) +* ∀ ≥ ∫ ⊂ Σ ln P() σ A ℤ Ψ
Path: / eq / piecewise / piece / geq
Done

$$f = \left\{ \begin{array}{lll} x_1 & \text{if} & x < 0 \\ x_2 & \text{if} & x \geq 0 \\ \square & \text{if} & \square \\ \square & & \text{otherwise} \end{array} \right]$$

Presentation MathML


Math Editor Page
f(x) +* ∀ ≥ ∫ ⊂ Σ ln P() σ A ℤ Ψ
Path: / mrow / mfenced / mtable / mtr / mtd / mrow
Done

$$f = \left\{ \begin{array}{lll} x_1 & \text{if} & x < 0 \\ x_2 & \text{if} & x > 0 \\ \square & \square & \text{otherwise} \\ \square & \square & \end{array} , \square \square \right]$$

The editor supports creating and editing the Content Math subset while being able to navigate through Presentation MathML. Every thing that is entered into the Editor is entered as Content Math. For example, entering `a*x^2+b*x+c=0` will be translated as the variable a times x to the power of 2 and added to b times x ...

Cursor

The Math Editor can be used entirely from the keyboard (See Keyboard Input). The cursor can be in one of four places. Either it is editing a variable or number, editing an empty block of text, next to a complicated expression, or has selected an expression. In each of these places there are several things that can be done.

Editing a variable, number, or block
At this point, the cursor is surrounded by a blue box and the user can type in expressions or even paste existing MathML. The expression will be parsed as soon as the cursor leaves the box or presses the Enter key (in the case of an expression) or immediately when MathML is pasted in. The user can leave the box by pressing clicking on the toolbar or by pressing the Left, Right, or Tab key. See Keyboard Input for more on expressions.

Next to a Complicated Expression
When a cursor is next to a complicated expression, the expression is shown with a light gray background (See Context). From this point, one of three things may be done. The user may add on to the expression. This is done by just typing. For example, if the cursor is to the left of $(-\pi)i$, the user may type `-1=e^` and parse the expression to yield $-1 = e^{(-\pi)i}$

One can select the expression by either pressing Shift+Right/Left (depending on whether the cursor is before or after the element), Delete, or Backspace key. See Selection for what can be done next.

Selection
When an expression is selected, several things can be done:

- Pressing the Delete or Backspace key will remove it
- Pressing Ctrl+X/C will cut/copy it
- Pressing Ctrl+V will replace the selection with the contents of the clipboard
- Clicking an item in the toolbar will replace the selected item

Context

Instead of using parentheses to denote which operations are grouped, the math editor highlights the current context for the operation. The context shows the position of the cursor relative to existing math in the editing area and is displayed using a gray background. An example of a confusing position can be shown using the following example. Suppose the editor contains the term $a + bc$ and the cursor is just after the c. If the user enters "^2" it is not clear what should be squared. At that position the user may want to square c, bc, or the entire term $a + bc$. This produces very different math, namely $a + bc^2$, $a + (bc)^2$, and $(a + bc)^2$. In the above example, the context would highlight precisely the math that ended up being in parentheses. One can think of the context as defining where the parentheses should go once the new math is entered.

Keyboard Input

There are several places the user can enter text into the editor. Most of them behave the same way, but listed below are common uses and specifics:
Common for all Text Entry Points

- Pressing the Enter key or moving the cursor out of the text box (by pressing the Left/Right, Tab key, or clicking elsewhere) after entering will cause the Math to be parsed.
- If the text cannot be converted to Math, it will appear with a red dashed line beneath it (like a spelling error) and must be corrected before saving.
- Simple algebraic expressions, logic operations, trigonometric functions, and subscripts can be entered and will be converted into math.
- If a shorthand notation exists for an operation, it will show up in the toolbar next to the name of the operation (See Toolbar. Shorthand notation is usually more natural (the operation, like addition, is between its arguments, like `a+2`
- If a shorthand notation does not exist for an operation, one can still enter the operation using the keyboard by typing the name of the operation which is also found in the menu (See Toolbar)

Categories
There are three categories of key presses and are enumerated in the table below.

- Shortcuts are preceded by pressing the Ctrl key (or the ⌘ key on Apple computers)
- Navigation keys move the cursor through the math

- Modification keys change the math in some way

Category	**Key**	**Condition**	**Action**
Ctrl+ (Apple ⌘+)	X	Math is selected	Cuts the selected Math to the clipboard and replaces it with an empty block (that can be deleted)
	C	Math is selected	Copies the selected Math to the clipboard
	V	Math is selected	Pastes MathML from the Clipboard (can be from other sources)
	Z		Undoes one step in the editor
	Y	Ctrl+Z was just pressed	Redoes one step in the editor
	E		Opens full-source editing
Navigation	Tab Shift+Tab		Moves to the next/previous free block

Category	Key	Condition	Action
	Left / Right		Moves to the previous/next element in the Math
	Shift+Left / Shift+Right	After / Before the Context	Selects the Context element (right next to the cursor)
		Before / After the Context	Selects the Context's parent
Modification	Enter		Attempts to parse the text entered next to the cursor
	Delete / Backspace	Cursor next to Math	Selects the Math Node (subsequent delete will remove the math)
		Math selected	Removes the node and replaces it with an empty block (a second press will remove the block as well)
		Cursor in block	Removes the empty block if it is allowed in MathML (in "a+b+c" any one variable can be removed, but addition requires at least 2 things to add)

Type	Input	Math Output
Calculator	`a*x^2+b*x+c=1/2`	$ax^2 + bx + c = \frac{1}{2}$
	`a && b || c != a -> b`	$a \wedge (b \vee c) \neq a \rightarrow b$
	`sin(x)^2+cos(x)^2=1`	$\sin^2(x) + \cos^2(x) = 1$
	`x_1+x_2<x_3`	$x_1 + x_2 < x_3$
Templates	`sum=n*(n-1)/2`	$\sum \square = \frac{n(n-1)}{2}$
MathML	`<pi />`	π
	`&#1207x;`	ҷ

Text Input Examples

Text Entry

This is a text entry place. See shortcuts. can paste MathML (Ctrl+V from Mathematica, MathType, etc), or enter simple algebra (see Shortcuts). Moving away using Enter, Tab, Left, Right will cause the input to be parsed and converted into Math.

Blocks

Blocks are holes that may need to be filled. (Click or Tab to them). Required blocks have a yellow background and optional ones are transparent and have a dotted border.

Click, double-click, highlight, (only right-click inside a text box)

Nuances / Limitations

There are several nuances in the editor, and common ones are listed here, along with workarounds. Also listed are limitations of the editor and things we'd like to get working soon.

- If more than two things are added or summed together, one cannot select only a subset of them.
- One cannot easily change a "+" sign to "*". To do this, you will need to copy the entire "+" operation and paste it, then remove the unwanted children.
- Moving children around by dragging is not possible. Unfortunately, this currently requires copying and pasting to the clipboard.

Limitations

Unable to change the domain of operations like Sum, Max, and Integrals. Operations like Sum, Max and Integrals may be over an interval, or when a certain condition holds (like $x \in$). The math editor allows editing these variations but does not always offer a way to create new operations. Currently, this must be done by hand by switching to the source edit view and manually replacing the `<interval/>` with a `<condition/>`.

Wrapping Math with Math

Sometimes it is necessary to add to existing mathematical operations. For example, adding higher terms to a polynomial. This can be done either by using the keyboard or with help of the toolbar. In the explanations below we start with "`b*x+c=0`" and create $ax^2+bx+c=0$

Keyboard Only
To add the ax^2 term:

Move the cursor to the left of $bx+c$ but make sure the context is **only** around $bx+c$ and that $bx+c$ is **not** selected. This can be done by clicking the "+" sign.
Enter "`a*x^2+`" (without the quotes) and press the Enter key.

Toolbar
Using only the toolbar to insert math is a bit more difficult because the editor does not infer multiplication or addition when pasting right next to existing math. We will need to "wrap" the existing math with the combiner operation (usually +,*, or ^) and then add in the new math.

Select $bx+c$ but make sure **only** $bx+c$ is selected. This can be done by double clicking the "+" sign.
Cut the existing math. This should create an empty block.
From the toolbar select the combiner operation. This should create at least one empty block.
Paste the math that was cut earlier into one of the empty blocks.
Select another empty block.
From the toolbar, insert the operation.

The Basics
This module provides a basic user's guide to Connexions' MathML editor. In it you will learn how to access the editor, how to create, edit, and delete MathML expressions using the editor, and how to insert the MathML code into a module.

Overview

This module covers some basic points of Connexions' MathML editor, now in beta testing. It is organized as a tutorial and will lead you through basic usage of the editor, though you can skip around the sections if you wish. The tutorial covers:

- how to access and use the editor
- how to create MathML expressions
- how to navigate through the structure of an expression
- how to edit expressions
- how to insert expressions created in the editor into a module
- how to delete expressions
- how to use the menu buttons to create and edit expressions

You will learn how to create the expression $ax^2 + bx + c$ using keyboard input and also using the toolbar menus.

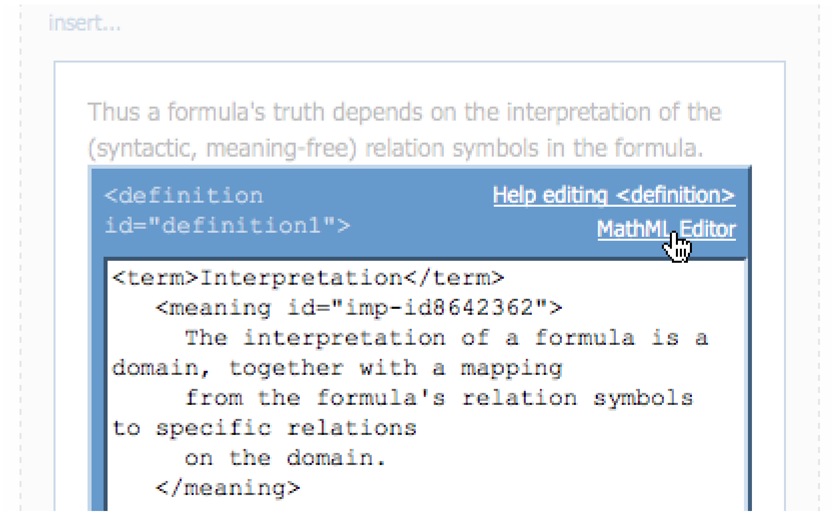
You will also learn about the basic layout of the MathML editor, the context of your cursor within the editing area, and how to select or highlight part or all of the MathML within the editor with key strokes or the mouse.

All of these aspects of the editor and more are covered in more detail in the MathML Editor: Manual.

Accessing the editor

Currently, the MathML editor is only supported in Mozilla Firefox. If you use Internet Explorer or another browser, you can download Firefox for free under the link.

When editing a module using Mozilla's Firefox browser, click on an element within the module to open a blue editing box.



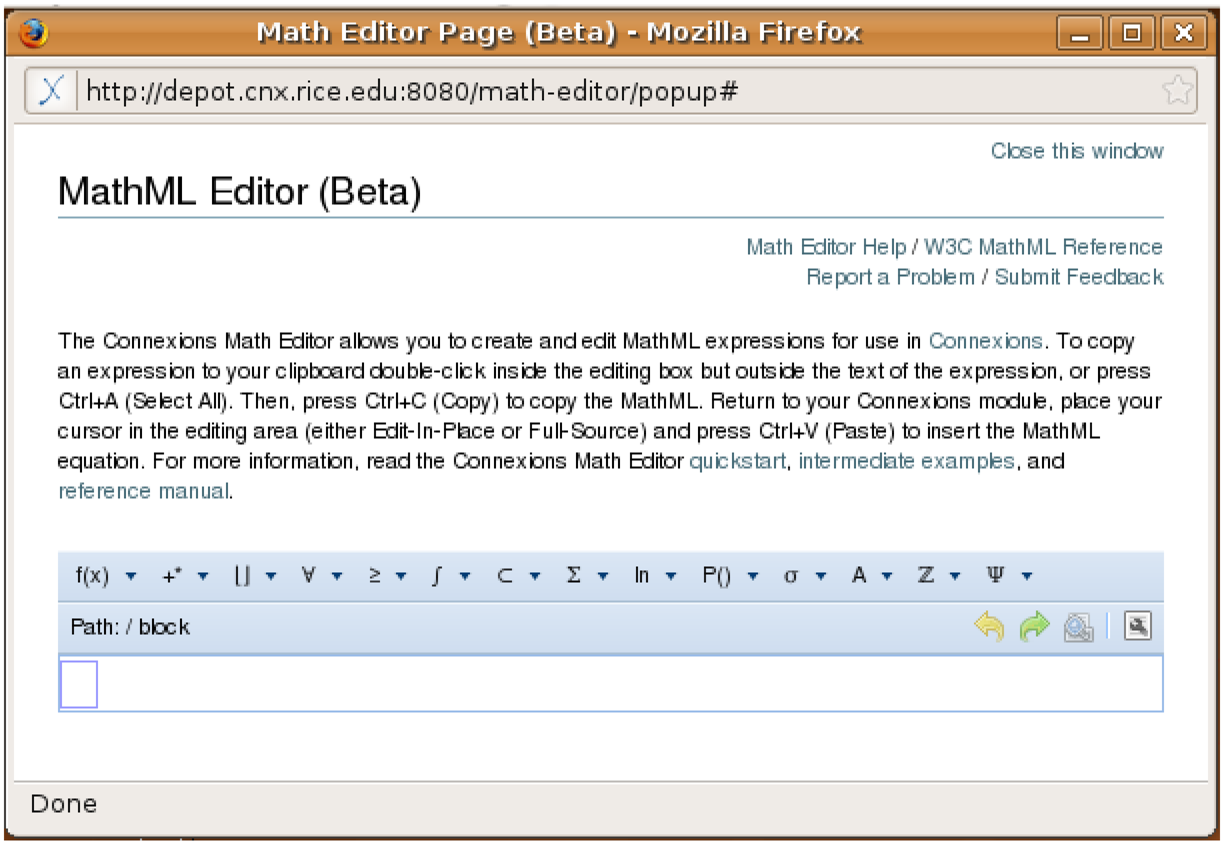
Thus a formula's truth depends on the interpretation of the (syntactic, meaning-free) relation symbols in the formula.

```
<definition
id="definition1">
```

```
<term>Interpretation</term>
   <meaning id="imp-id8642362">
     The interpretation of a formula is a
domain, together with a mapping
     from the formula's relation symbols
to specific relations
     on the domain.
   </meaning>
```

Begin editing and in the top-right corner is a MathML Editor link

On the top-right hand side of the box is a "MathML Editor" link which will open up the editor.


Math Editor Page (Beta) - Mozilla Firefox

http://depot.cnx.rice.edu:8080/math-editor/popup#

Close this window

MathML Editor (Beta)

Math Editor Help / W3C MathML Reference
Report a Problem / Submit Feedback

The Connexions Math Editor allows you to create and edit MathML expressions for use in Connexions. To copy an expression to your clipboard double-click inside the editing box but outside the text of the expression, or press Ctrl+A (Select All). Then, press Ctrl+C (Copy) to copy the MathML. Return to your Connexions module, place your cursor in the editing area (either Edit-In-Place or Full-Source) and press Ctrl+V (Paste) to insert the MathML equation. For more information, read the Connexions Math Editor quickstart, intermediate examples, and reference manual.

Once clicked, a popup window will appear containing the Math Editor

The popup window can remain open while editing a module and can even stay open while editing several modules. At any point one can close the window, but the contents of the editor will be lost.

The editor has two main sections: the toolbar and the main editing area.

The toolbar consists of

- a Menu containing mathematical operators
- the Path bar to help you find the context of your cursor
- buttons for Undo/Redo
- a Preview button, which will display the math as it will look in a published module
- and a View Source button, which displays the raw MathML code

The editing area is the main area for creating math. It begins empty, but math can be pasted directly in here from Connexions, or created on the spot by either typing in expressions or using the toolbars. See the section Inserting expressions into a module below for how to move math from Connexions to the math editor and back.

Creating expressions

When you first open the MathML editor, the Editing area should have one blank box in it, called a "block". A block is an empty slot that can be filled with a number, variable, or longer expression consisting of at least one operator and its associated arguments.

Click inside of it to place your cursor inside the empty block.

There are two ways to insert math using the keyboard. The simplest way is to use keyboard shorthand notations for operators. These exist for simple algebra notations, and are similar to the buttons on a calculator (for instance, `*` for multiplication, `-` for negative or minus). A full list of Keyboard shortcuts is located here.

The MathML editor will correctly parse a sequence of text and numbers and keyboard shortcuts into operators and arguments. For instance, `3-4` will be interpreted as "3 minus 4", and `3/(x+9)` will be interpreted as "3 divided by the quantity x plus 9".

The second way to create operations using the keyboard is to type out the MathML name. This is analogous to choosing the operation from the toolbar menus and is discussed in the same section.

An example

With a blank MathML editor open, try typing in the following (or copying it from here and pasting into the editor):

```
x^2+b*x
```

Hit `Enter`. The MathML editor will display this as:

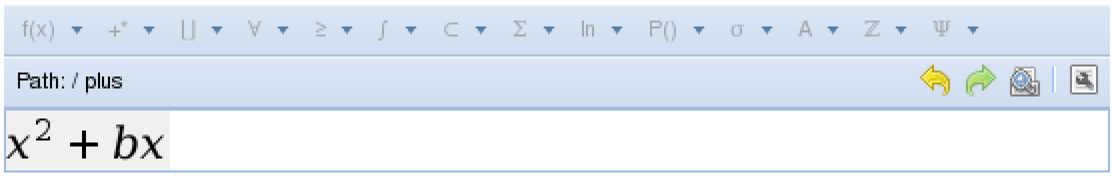
$x^2 + bx$

Notice that we had to be explicit about our operations. Although most textbooks, and even our MathML editor, represents the product visually by printing b and x next to one another, we had to explicitly mark the multiplication. The editor will display the `times` operation in different ways, depending on the surrounding operations.

If you try to represent the product of b and x with `bx`, that portion of your expression will be highlighted, indicating that there is a problem with the code that must be fixed before being used in a module. Simply click on bx and replace it with `b*x`.

Navigating through an expression

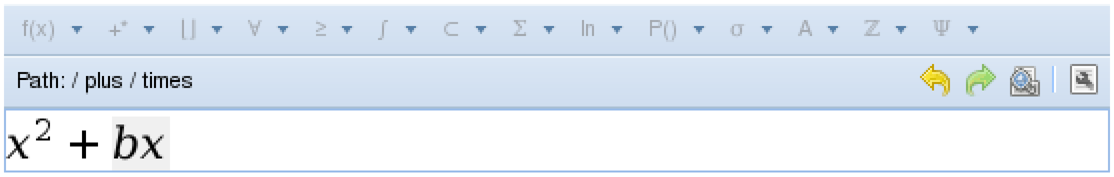
If you have just created an expression and pressed `Enter`, tight now your cursor should be at the right edge of your expression. If it isn't, click somewhere within your expression, and press the right arrow key until the cursor no longer moves and the path no longer changes in the Path bar. In this position, the Path bar should display `Path: / plus` and the entire expression should have a light grey background.



The context of the cursor is the entire plus operation.

Press the left arrow key once. The cursor should still be on the right-hand side of the expression, but the Path bar should read: `Path: / plus /`

`times` and now only the bx term should have a light grey background.

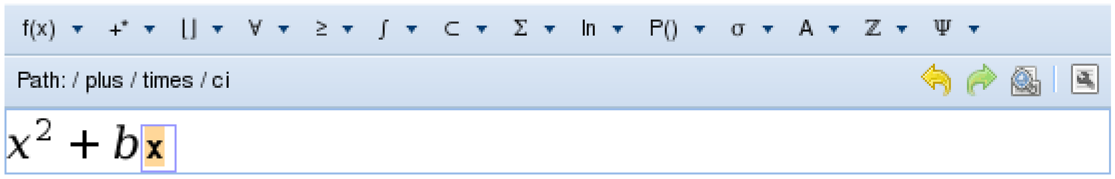


Path: / plus / times

$x^2 + bx$

The context of the cursor is thetimes operation.

Press the left arrow again. Now the x should appear within a block and be selected. The path will change again to `Path: / plus / times / ci` The term `ci` represents an "identifier", in this case our variable x.



Path: / plus / times / ci

$x^2 + bx$

The context of the cursor is the ci block.

Continue pressing the left arrow key across your expression to see what happens. Notice that once you reach the left side of the b, the path should again read `Path: / plus / times` and the product bx should have a light grey background. When you reach the far left side of the expression, the path should again read `Path: / plus` and the entire expression should have a light grey background.

What is going on here?

MathML treats operators and their arguments as nested elements. In our example of $x^2 + bx$, b and x are both arguments of the operation called `times`. The product "b times x" is just one argument within the `plus` operation. The other argument of the `plus` operation is of course x^2. This `power` operation itself has two arguments: x and 2. In this way, b is more closely bound to x than it is to either argument of the `power` operator.

We could picture these operators and arguments in a parenthetical structure: `( ( (x)^(2) ) + ( (b)*(x) ) )` The parentheses group arguments of each operator together. A set of parentheses encloses each number or variable separately, and a set of parentheses encloses each entire operation - for instance, the entire product bx or the entire sum $x^2 + bx$.

This parenthetical structure can also be visualized as a tree-like structure. See the MathML Editor: Manual for an example.

By moving our cursor left and right, we move deeper in or out of these parenthetical or tree-like structures, represented by a longer path name, and a smaller unit of shaded math. The Path and shading lets us know the context of our cursor, i.e. where within the parenthetical or tree-like structure our cursor is. Essentially, navigating to different contexts of the MathML expression lets us place new expressions anywhere we want, as we'll show below.

Editing expressions

Adding an argument to an existing operation

Be sure the editing area of the MathML editor contains the expression we entered, $x^2 + bx$. Let's change this to $x^2 + bx + c$.

To do this, we will have to add an argument to the addition operation. Move your cursor to the end of the expression by clicking on the last x and then pushing the right arrow key a couple of times to move the cursor to the right of the `times` operation. Then type in `+c` and press `Enter`.

You do not always have to add on to the right-hand side of an expression. Try moving your cursor to the beginning of the expression, typing in `c+` followed by `Enter`. Experiment with adding more arguments to the `plus` operation just after the `power` operation or just before the `times` operation.

Creating a new operation

In our example, $x^2 + bx + c$, let's change x^2 to ax^2. We are not adding an argument to an existing operation, because x and 2 are arguments of a `power` operator, while we want to make the entire `power` operation one argument of a `times` operation. The other argument will be the new element that we are adding, a.

The underlying structure of our new expression will look something like this: `( ( (a) ( (x)^(2) ) ) + ( (b)*(x) ) + (c) )`

For this we will need to choose the context of our cursor carefully, or we will get different math. Make sure your cursor is on the left hand side of the `power` operation, the path says `Path: / plus / power` and that x^2 has a light grey background. This means the context of the cursor is the expression x^2, which is good. We only want to multiply x^2 by a, nothing more and nothing less.

Then type in `a*` and press `Enter`. The MathML should display as:

$$ax^2 + bx + c$$

What if our cursor had been in a different context when we typed in `a*`? We could have moved our cursor farther to the left so that the context was the entire `plus` operation, where the path bar would read `Path: / plus` and the entire expression would have had a light grey background. In this case, typing in `a*` would have affected the entire grey area and would have resulted in this display:

$$a\left(x^2 + bx + c\right)$$

If our cursor was next to the x within the x^2 term, a block would appear around the x and the path bar would read `Path: / plus / power / ci`. Typing in `a*` next to the x in the block would mean that now the quantity ax would be squared and would result in this display:

$$(ax)^2 + bx + c$$

Inserting expressions into a module

Let's put our expression $ax^2 + bx + c$ into a module we're editing.

The MathML editor won't automatically insert your MathML expression into a module. You will have to copy and paste the MathML code. You can paste it into any module you wish, not only the module that you first accessed the MathML popup window from. The popup window can remain open even if you edit several different modules, but once you close it all contents of the editor will be lost.

You can press `Ctrl+A` (or `⌘+A` on Macs) to select your entire expression. You can also double-click in the white box outside of your expression to select it all. If you want to select only a portion of the expression, hold the `Shift` key down and use the left and right arrow buttons, or double click on the operator of just the portion you want to select (i.e. double click `+` to select everything within the `plus` operator, or double click on two multiplied objects to just select that product).

Copy your expression by pressing `Ctrl+C` (`⌘+C`). Open your module and click on a segment to edit. Press `Ctrl+V` (`⌘+V`) to paste the MathML code directly into your module.

Why can't I select the MathML with my mouse?

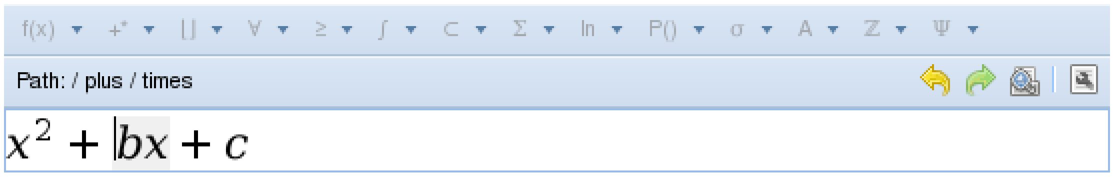
There are two ways to select expressions within the editor. By using one of the methods above, you copy the underlying MathML code. This is what you need to copy MathML code over to a module, or to select a portion of your code to delete.

By clicking and dragging over an expression within the edit box, you select only the text. Like any other text you enter, if you copy part of your code and paste it as an argument of an operator within the editing area, the MathML editor will correctly parse it and generate the underlying code. This is also called "wrapping" your existing math.

Deleting expressions

There are several ways to delete part or all of your expression. You can either move your cursor to the context of the portion of your expression you want to delete, then push `Delete`, or you can select the portion of your code you want to delete, and press `Delete`.

To delete bx from our expression $ax^2 + bx + c$, move your cursor until the context is only bx. The bx portion of the expression should have a light grey background, and the path bar should read `Path: / plus / times`.

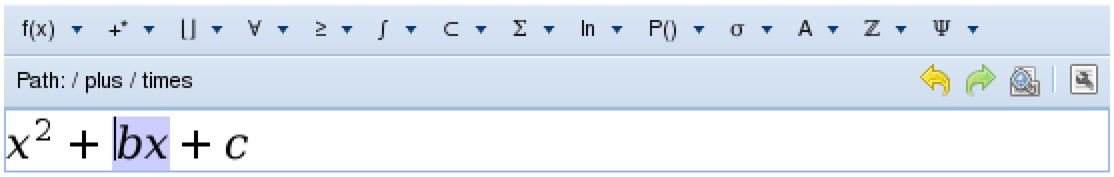


Path: / plus / times

$x^2 + bx + c$

The context of the cursor is the times operation.

Press `Delete` once, and that part of your expression should be highlighted in blue.

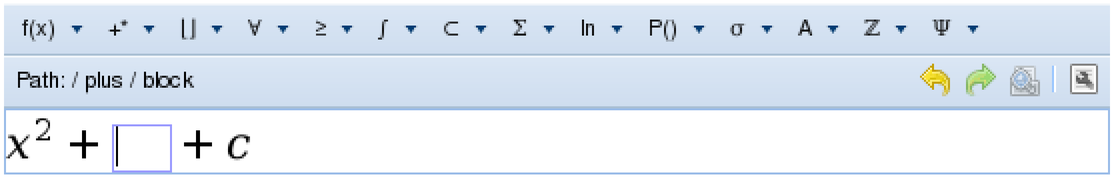


Path: / plus / times

$x^2 + bx + c$

The times operation is highlighted.

Press `Delete` again and that portion will delete.



Path: / plus / block

$x^2 + \boxed{} + c$

The cursor will move to the next argument.

You should be left with a blank block where the expression bx used to be. By clicking `Delete` you have deleted the content of one argument of the `plus` operator, but not the argument itself. You can type in a different expression that will replace bx, or if you simply want the expression $ax^2 + c$, press `Delete` one more time to delete the block. Notice that the context automatically moves to the next argument, which in our case is the `ci` element c.

Creating expressions with the toolbar menus

The toolbar menus are only activated when something is selected within the editing box. This can be either an entire operator, like the expression bx, or it could be just one argument of an operator, like b or an empty block. Since the editing box contains only an empty block when you first open it, the block is automatically highlighted and the toolbar menus are accessible.

However, note that if you select a portion of your MathML expression and then insert an operator from the toolbar, the selected portion will be overwritten. The operator will **not** appear behind or in front of the selected expression. New users are often confused by this behavior. Essentially, keyboard input and the toolbar function in different ways.

Remember that each operator in MathML has a specific number of arguments that can be associated with it. For instance, the operator `plus` must have at least two arguments (as in $x + y$) up to an unlimited number of arguments (as in $1 + 2 + 3 + 4 + 5 + 6$...). The operator `root` has one required argument (the radicand, as in $\sqrt{3}$), as well as one optional argument for specifying which root power (as in $\sqrt[6]{3}$).

When you insert a series of characters into the editor, the editor will correctly parse some of the symbols you enter as operators, and some as arguments of the operators. Furthermore, it correctly determines **which** operator you have typed in, and correctly associates the surrounding symbols as arguments of that operator.

The menu buttons in the toolbar will paste an operator over your highlighted expression, and insert blank blocks for all the required and optional arguments of the operator. This means that keyboard input and toolbar insertion are useful for different things.
The toolbar is recommended:

- When you are just beginning a new expression and already know the relations of the arguments to one another. For instance, if you have a written version of an equation in front of you and want to recreate it in the MathML editor so that you can paste the MathML code into a module. We recreate the equation $ax^2 + bx + c$ in the sections below using only the toolbar, or find a more complicated and in-depth example in the module MathML Editor: Intermediate.
- When the operator has many arguments associated with it.

- When the operator does not have a keyboard shortcut, or if you do not know the keyboard shortcut. To the right of each menu entry, there is the keyboard shortcut for that operator. For complicated operators, you often must type in the name of the element (for instance, "root"), and the editor will provide a visual representation (the radical sign, in this case) and blank boxes for argument entry.

Keyboard input is recommended:

- For most simple MathML entry.
- When you want your existing expression in the editing area to become one argument of a new operation. To add a new argument to the plus operator $x + y$, you need only type `+z`. The editor interprets z as one argument of the `plus` operator, and the previously existing expression as the other argument. Toolbar insertion cannot do this. Some examples of workarounds are given below.

The following sections will show you how to create an expression from scratch using mainly toolbar entry, and how to use part or all of your existing expression as one argument of a new operator.

Creating a new expression

The toolbars are useful when creating new expressions because you work "from the top down". The operator with the broadest scope is inserted from the menus first. Then, the arguments are filled in with variables and numbers, or with another complex expression. If one complex expression is itself an argument of an operator, you can use the toolbar to insert the operator first, and then fill in the arguments again. Continue this iteration with successively narrower scoped operators, until only number and variables are left to be inserted as arguments.

Let's create $ax^2 + bx + c$ using the toolbar menus, and in the same order that we created it with keyboard input above. We'll first insert $x^2 + bx$, then add c as an argument to the existing `plus` operator, and finally add a as an argument in a new operation.

Make sure the editing area in the MathML editor is blank. It should already be blank if you've just opened it. If you have some text entered, delete it first.

The expressions x^2 and bx stand in relation to one another as arguments of a `plus` operator, so we will start by inserting a `plus` operator and then inserting successively smaller chunks of the expression.

Click the `plus` operator from the toolbar.
To create the x^2 expression, click within the first block in the `plus` operation, then select the `power` operator from the toolbar.
Click within the first block in the `power` operation, and type in x.
Click within the second block in the `power` operation, and type in 2.
To create the bx expression, click within the remaining block, which should be the second argument of the `plus` operation, and select the `times` operator from the toolbar.
Click within the first block of the `times` operation, and type in b.
Click within the second block of the `times` operation, and type in x.

Adding an argument to an existing operation

There are two ways we can add the final argument c to the `plus` operator. One way would have been to add an extra argument to the `plus` operation immediately after we inserted it from the toolbar menu. After Step 1 above, click in either blank block on each side of the `+` sign. Then insert another `plus` operation. Essentially we are inserting the entire operator as an argument of the first `plus` operator, but the editor will correctly condense this and turn all three blocks into arguments of one `plus` operator. The same thing occurs with keyboard input when, say, you replace the 6 in $6 + 7$ with $5 + 6$.

The second method would be to want to add a new argument to the `plus` operator after we already created the expression $x^2 + bx$. In this case, we essentially want to make $x^2 + bx$ the first argument of a new `plus` operator, and then add c as the second operator. This is known as "wrapping" existing math in a new operation.

Select the entire expression and cut it by pressing `Ctrl+X` (or `⌘+X` on a Mac), so that you are left with a blank editing area.
Select the `plus` operator from the toolbar menu.
Paste in the expression $x^2 + bx$ as the first argument of the operation.
Click in the remaining block and type in `c`.

Creating a new operation

You will also have to wrap existing math when inserting a new operation. Let's change the first expression x^2 to ax^2.

Select and cut the `power` operation x^2. You should be left with a blank block + $bx + c$.
Insert a `times` operator from the toolbar menu.
Click on the first block of the times operator and type in `a`.
Click on the second block of the times operator and paste in the expression that you cut.

Tutorials and Examples

Introduction

This module presents step-by-step instructions for creating, and then editing well-known formulas to illustrate how to use the editor. The example and what properties of the editor it illustrates are listed below. All of these assume you have a blank Math Editor open.

- Quadratic Equation (Simple text entry, Special characters)
- Quadratic Equation using mostly the mouse
- Advanced Text Entry

Quadratic Equation

As a simple example, we'll step through several ways of writing the well-known Quadratic Equation (with real or complex coefficients):

$$ax^2 + bx + c = a\left(x - \frac{-b + \sqrt{b^2 - 4ac}}{2a}\right)\left(x - \frac{(-b) - \sqrt{b^2 - 4ac}}{2a}\right)$$

Method 1: Pure Keyboard

Probably the quickest way to enter math is by using the keyboard. This method requires entering a total of 3 statements and a few Tab key presses.

Start off with a blank editor.
Enter the following into the main editing area "`a*x^2+b*x+c=a*(x-(-b+root)/(2*a))*(x-(-b+root)/(2*a))`". See details below for
Press the Enter key. This will cause the text to be parsed and converted into math.

Most of the text in step 2 should look similar to the notation used in calculators, except for "`root`". Many calculators follow different conventions for entering complicated math operations like integrals and vectors. For this version of the editor we decided to wait for feedback from users on which convention to adopt. Until one is chosen, any math element defined in the W3C MathML Specification can be entered. The toolbar also provides a way to see the available commands.

Finish Entering Equation
At this point the editor should have 2 remaining boxes that need to be filled out, and 2 optional ones (the degree of the root). To fill in the rest, you will need to do the following:

1. Press Shift+Tab four times to move to the first empty block. (That is, hold down the Shift key, press the Tab key, and release the Shift key four times).
2. Enter "`b^2-4*a*c` " into the empty block under the radical.
3. Press the Enter key to convert the input into math.
4. Press Shift+Left arrow key to select $b^2 - 4ac$
5. Press Ctrl+C to copy the selection to the clipboard.
6. Press the Tab key twice to move to the other empty block.
7. Press Ctrl+V to paste the selection into the current block.

Now, the equation should be complete. In the previous steps we used the Tab key to navigate to empty blocks that still needed information in them, skipping over optional ones. We used Shift+ arrow keys to select math and Ctrl+C and Ctrl+V to copy and paste that math.

Paste into Connexions
Finally, we need to copy the math and paste it back into a module. We already used the same technique above. Right now, the cursor should be just to the right of the second $b^2 - 4ac$. The following steps will place the newly created quadratic equation back into the Connexions module.

Press Ctrl+A to select the entire formula, or Shift+Right (or Shift+Left) until the math you want to copy is selected.
Press Ctrl+C to copy it to the clipboard.
Switch back to the window where you were editing the module.
Place the cursor at the location you want to insert the quadratic formula.
Press Ctrl+V to insert the formula.

Summary
In this tutorial we entered the quadratic equation entirely through the keyboard. We used the Tab and arrow keys to navigate through math content, the Return key to convert input text into math, and Ctrl+C and Ctrl+V to copy and paste both within the editor and between the editor and the main module editor.

Next, we will do the same example using the mouse and toolbar.

Method 2: Toolbar

This method requires a bit more time because we will need to click the toolbar for every character (like "+", "*", or "/" in the previous method). Instead of doing the entire equation, this tutorial will step through creating only part of it: $\frac{-b+\sqrt{b^2-4ac}}{2a}$

Start off with a blank editor.
Click the "Arithmetic" category in the toolbar.
Click the "Divide" operation in the menu.
Click the top empty block (numerator).
Click the "Arithmetic" category in the toolbar and the "Plus" operation in the menu.
Click the left empty block.
Click the "Arithmetic" category and the "Negate" operation in the menu.
Enter "b" in the top-left empty block
Click the other empty block in the numerator.
Click the "Arithmetic" category and the "Root" operation in the menu.
Click the empty block under the radical.
Click the "Arithmetic" category and the "Minus" operation in the menu.
Click the left empty block.
Click the "Arithmetic" category and the "Power" operation in the menu.
Enter "b" and "2".
...

Using the toolbar is a bit more tedious, but serves as a way to find operations that can be expressed in MathML. Some operations have variations (A sum can take a variable and limits, or a variable and a condition) but see Limitations on how to enter them in.

Paste into Connexions
Pasting the Math back into Connexions can be done the same way as before, or can be done via the Edit menu in the browser. Again, we must select the entire equation. This can be done by highlighting the equation using the mouse, or double-clicking the division bar (since it is the outer-most operation). Once highlighted, you can Click Edit, and either Cut or Copy from the main browser menu bar. If you switch back to the Connexions module editor, you can Click Edit and then Paste again from the menu to paste the newly created math back into a module.

Summary
In this tutorial we entered a part of the quadratic equation using the mouse and toolbar buttons. We used the mouse to select move the cursor and select math, the toolbar to insert new operations, and the browser's Edit menu to copy and paste math between the editor and the main module editor.

Next, we will discuss some more advanced math editing.

Advanced Editing

So far we've gone through creating math from scratch. In this section, we will look at how to insert more elaborate symbols, change how variables look (Presentation MathML), and customize some of the operations provided in the toolbar.

Elaborate Symbols
So far we've used simple characters available from the keyboard. The quadratic formula is frequently written with a plus-minus sign like:

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

In order to get this, we will need a little bit of Presentation MathML. This is because plus-minus is not an operation represented in Content MathML.

We will start with one part of the formula (asdescribed above)
To save some time, copy the $\sqrt{b^2 - 4ac}$into the clipboard.
Highlight everything in the numerator (it should be the entire plus operation)
Replace it with "`mrow`" and press the enter key
Enter "`-b`" and move to the next block
Enter "`±`" and move to the next block
Paste the part of the formula we copied earlier

`mrow` is used to control how Math is displayed to the user. In this case we used it to insert a plus-minus symbol between $-b$ and $\sqrt{b^2 - 4ac}$. The Unicode standard defines many characters but the Unicode Mathematical Operators document may be a useful reference.

Customize the Look of Variables
There are many elaborate ways to customize how a variable looks. These are defined in the W3C MathML Specification . We will list off a few common ways to customize.

- Subscripts like x_i can be entered by typing "`x_i` " or using "`msub` "
- A variable with both subscripts and superscripts can be entered using "`msubsup` "
- Backets like $(-\infty, 0]$ can be added using "`mfenced` " and then changing the symbol used for the open and close bracket (by editing the source).
- Unlike subscripts which place content above/below and to the right, "`munderover` " places math directly above or below.
- A table can be added using "`mtable` "

Customize Toolbar Operations
Many operations that operate on a range have several ways of specifying the range they work on. For example, the following are equivalent:

$$\overset{n}{\underset{i\quad 1}{}}\ i^2$$

$$ {}_{i \in S}\, i^2 \quad S = \{i|\ (i > 0)\ \wedge\ (i \leq n)\}$$

Changing the range these operations required switching to the MathML source and being familiar with the W3C MathML Specification . To change the former to the latter, we start with a clean "`sum` " operation. Then, to decrease the amount of hand editing, we can type "`i in S` " to the right of the equal sign. Then, we switch to MathML Source and replace every occurrence of "`interval` " with "`condition` " and removing the special "`<block ...>` " element just above the `</condition>` .

W3C MathML Specification

The W3C MathML Specification defines all math used in Connexions modules. It provides ways to represent formulas in a way that records the semantic meaning in the formula (Content MathML) as well as a way to lay out variables and formulas (Presentation MathML).